# EVALUATING CONTINUOUS-TIME SLAM USING A PREDEFINED TRAJECTORY PROVIDED BY A ROBOTIC ARM

Bertram Koch, Robin Leblebici, Angel Martell, Sven Jörissen, Klaus Schilling, and Andreas Nüchter

Informatics VII – Robotics and Telematics, Julius-Maximilians University Würzburg, Germany
andreas@nuechti.de

**Commission III WG III/2**

**KEY WORDS:** 3D laser scanning, SLAM, continuous trajectory optimization, ground truth, evaluation

**ABSTRACT:**

Recently published approaches to SLAM algorithms process laser sensor measurements and output a map as a point cloud of the environment. Often the actual precision of the map remains unclear, since SLAM algorithms apply local improvements to the resulting map. Unfortunately, it is not trivial to compare the performance of SLAM algorithms objectively, especially without an accurate ground truth. This paper presents a novel benchmarking technique that allows to compare a precise map generated with an accurate ground truth trajectory to a map with a manipulated trajectory which was distorted by different forms of noise. The accurate ground truth is acquired by mounting a laser scanner on an industrial robotic arm. The robotic arm is moved on a predefined path while the position and orientation of the end-effector tool are monitored. During this process the 2D profile measurements of the laser scanner are recorded in six degrees of freedom and afterwards used to generate a precise point cloud of the test environment. For benchmarking, an offline continuous-time SLAM algorithm is subsequently applied to remove the inserted distortions. Finally, it is shown that the manipulated point cloud is reversible to its previous state and is slightly improved compared to the original version, since small errors that came into account by imprecise assumptions, sensor noise and calibration errors are removed as well.

## INTRODUCTION

Nowadays, there exist a wide variety of simultaneous localization and mapping (SLAM) algorithms for a lot of different applications. Online SLAM algorithms, such as Google's recently published cartographer (Hess et al., 2016), let laser scanner systems simultaneously localize in an unknown environment and generate high precision 2D or 3D maps of their surroundings in real-time. In contrast to this, offline SLAM provides a post-processing step for separate 3D point clouds. Recently, continuous-time SLAM approaches are used to optimize the trajectories acquired during mobile mapping (Barfoot et al., 2014; Anderson et al., 2015; Bosse et al., 2012; Lehtola et al., 2016; Zhang and Singh, 2014; Kaul et al., 2016). This is for example achieved by globally consistent scan matching to find an optimal alignment of 2D profiles to increase the inner accuracy of the point cloud. The benchmarking of such algorithms is an important key aspect to test their reliability and performance. To this end, a highly precise experimental setup has to be built to obtain an accurate ground truth. Distorting the ground truth trajectory yields a possibility to verify the performance of the algorithm by comparing the resulting point clouds.

This paper focuses on benchmarking continuous-time SLAM. We will show that inaccuracies of geometrical calibration and timing issues are counterbalanced to some degree. We will describe in detail the laser scanner measuring unit with its hardware and software components, which is then used to benchmark our continuous-time SLAM algorithm under different aspects.

### Continuous Time SLAM

The automatic registration of terrestrial laser scans is considered solved, e.g., by using natural features in projections as key points (Houshiar et al., 2015) in combination with the ICP algorithm (Besl and McKay, 1992). Its extension to globally consistent scan matching has also been presented (Nüchter et al., 2010). The later method creates a graph of overlapping scans and optimizes a global error function. However, localization of a mobile laser scanner (MLS) without using a global reference coordinate system, e.g., global navigation satellite system (GNSS), and with sensors attached only to the mobile scanner platform is one of the grand challenges in laser scanning research. Barfoot et al. (2014) and Anderson et al. (2015) used a regression to Gaussian processes to optimize the trajectory that their mobile mapping system has taken. Bosse et al. (2012) used a lightweight laser scanner and spring system to achieve a smooth motion for a handheld mapping system. Lehtola et al. (2016) built an experimental mobile mapping system based on a FARO scanner and a rolling wheel. Similarly, constantly rotating scanners were used in the setups of (Zhang and Singh, 2014; Kaul et al., 2016; Nüchter et al., 2015). The task of continuous Time SLAM is to deform or transform the trajectory, such that the quality of the point cloud is improved.

A solution to continuous time SLAM is needed for various applications, ranging from indoor mapping and personal laser scanning to autonomous driving. If the 3D scanner is fast, like for instance the Velodyne HDL-64E scanner, then using motion compensated 3D scans result in descent maps, when the point clouds are treated as separate point clouds in an Online or Offline SLAM solution (Moosmann and Stiller, 2011). In general, however, every 3D measurement must be corrected from the SLAM algorithm depending on its time-stamp.

### Evaluating SLAM

In outdoor scenarios, GNSS provides a sufficient precise reference for benchmarking the trajectory of the mapping systems. Furthermore, control points are used to evaluate the overall point cloud quality. Furthermore, using reference blocks are often used to check accuracy and repeatability.

Similar ideas are used in the robotics community to evaluate the result of SLAM algorithms. Schwertfeger et al. (2011) uses reference blocks, so-called fiducials to evaluate maps created in RoboCup. Later on, Schwertfeger and Birk (2013) scored topological map structures. Wulf et al. (2008) used an independently

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
doi:10.5194/isprs-annals-IV-2-W4-91-2017  

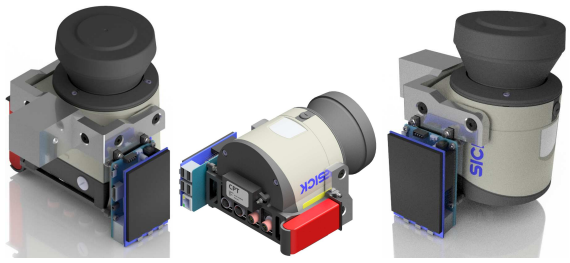

Figure 1. CAD model of assembled measuring unit combining: SICK LMS141 2D laser scanner, a single board computer and the necessary electrical components.

available, accurate environment map of an urban area and the Monte Carlo localization (MCL) technique that matches sensor data against the reference map in combination with manual quality control.

### 3DTK – The 3D Toolkit

*3DTK – The 3D Toolkit* (Andreas Nüchter et al., 2017) is an open-source toolkit for efficient processing of 3D scans and point clouds in general. It contains among other things algorithms and methods for point cloud registration, shape recognition, and for fast viewing of scenes. This paper uses the toolkit for developing the benchmarking methods.

## SYSTEM OVERVIEW

### Hardware Description

The laser profiler was attached to the robotic arm using a modified Schunk PT- AP-70 gripper and a custom designed mounting bracket seen in figure 1. The measuring unit with the laser scanner and all electrical components are bolted to the mounting bracket, which provides a mechanical fixture for the gripper to attach to. This setup enables to easily dismount the assembly and modify it. The bracket was a 3D printed prototype created out of polylactic acid (PLA) plastic already provided sufficient sturdiness. The measuring unit was designed to work independently and thus the assembly contains a LiPo battery, a DC/DC converter (12 V to 5 V) and a Raspberry PI 3 model B single board computer for data collection of the laser scanner data using the robot operating system ROS (Quigley et al., 2009).

**2.1.1 Robotic Arm** The deployed robotic arm is a KUKA KR16 industrial robot with six individual revolute joints. It is capable of very precisely traversing a trajectory and simultaneously change the orientation of the tool tip using the spherical wrist of the robot. This is needed for the laser scanner to obtain a full 360 °-view of its environment. For a more detailed description of the predefined trajectory see section 4 The actual position of the robot is monitored via network using the robot sensor interface (RSI) provided by KUKA. The robotic arm has a path accuracy of ±0.9 mm for linear motion and ±0.8 mm for circular motion with a repeatability of ±0.2 mm and ±0.4 mm respectively. This provides sufficient accuracy to receive a precise ground truth used to verify the continuous-time SLAM algorithm.

**2.1.2 Laser Scanner** The SICK LMS 141 security prime laser scanner is used for the measuring unit. It measures at a frequency of 50 Hz and has an operating range of 0.5 m to 40 m. The 2D distance measurements are a representation of the scanning plane in polar coordinates, with an angular resolution of 0.5 ° and a field of view of 270 °. Measurement data output is provided via Ethernet interface in real time. With a systematic error of ±30 mm and

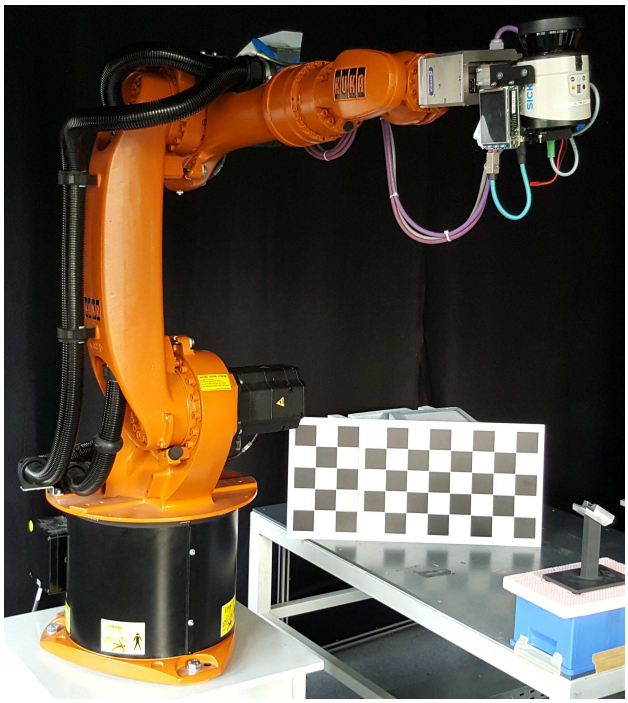

Figure 2. Overview of the test setup - Laser scanner attached to the KUKA KR 16 robotic arm.

a statistical error of ±12 mm up to 10 m range it is well suited for the purpose of creating a high precision reference point cloud of the environment.

### Software Description

The functionalities of the system are distributed into three main software components: Two ROS nodes for data collection and one application for data processing. ROS is designed to distribute the software components onto separate systems which communicate with each other using the native ROS publisher and subscriber protocol with predefined topics and message types. When this functionality is applied by connecting the recording nodes to the same network it acts as one system. One ROS node is used to collect the 2D laser scanner data (LMS1xx node) on the Raspberry Pi which is connected via WiFi. A second ROS node, the RSI handler, is implemented on a separated PC and used to acquire the joint angles of the robotic arm in real-time. The resulting trajectory of the tool position of the robotic arm is computed in the robot base coordinate system using the direct kinematic equations of the robotic arm. To ensure a precise timing between the two machines, the two devices are time-synchronized using the Network Time Protocol (NTP). In a consecutive step the data is combined to create a 3D reference point cloud as input for the continuous-time SLAM algorithm. The system components and its interactions are illustrated in Figure 3.

**2.2.1 LMS1xx Node** The LMS1xx package implements a basic ROS driver in C++ for the SICK LMS1xx line of LIDARs, which is used for the configuration of the laser scanner as well as the data acquisition. It collects a profile of 541 2D distance measurements in polar coordinates with reflectance value at a rate of 50 Hz and publishes them as *sensor_msgs/LaserScan* messages to the ROS network. The Raspberry Pi running the node is controlled via the ssh protocol over the network, simplifying the operation and data acquisition.

**2.2.2 RSI Handler Node** The RSI handler node is developed as a desktop application in Qt5. In the interface it is possible

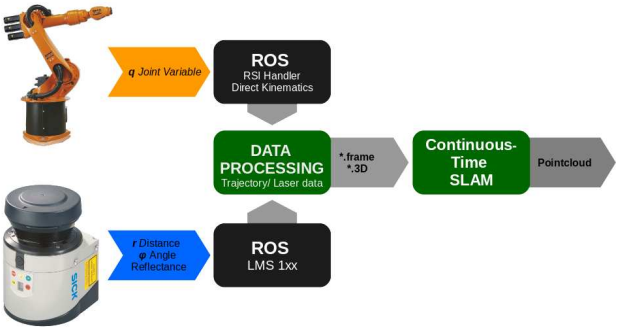


Figure 3. Data acquisition and processing pipeline.

to configure the listening port for data, if a multicast subscription to the network should be attempted and if the data should be published to a ROS topic. The KUKA system publishes, via its proprietary RSI API, the instantaneous joint angles for the robot with the current time stamp. This data is published through UDP to a dedicated computer in the robotics lab, which then relays the information to a multicast group defined by the network administrator. The KUKA RSI API publishes data every 12 ms. The RSI handler node resides in a computer that is a member of this multicast group and then listens for incoming data. When data is received, the direct kinematics for the tool position are solved with the kinematic parameters provided by the manufacturer, and this yields the homogeneous transformation matrix that is published in a ROS topic for recording. The transformation matrix contains the information for converting points in the frame of reference of the tool to the frame of reference of the base of the robot.

**2.2.3 3D Point Cloud Generation** The data from the LMS1xx node and the RSI handler node are collected in a single rosbag file on a separate device which solely acts as a listener in the ROS network. The resulting rosbag is then processed in the following steps: First, the data of interest is selected by detecting the start and stop point of the trajectory in the pose measurements of the robotic arm. The ROS time stamps are then used to find the corresponding laser scans in that time interval. The laser scans are exported in left-handed $x-y-z$ convention to be suitable for further processing with continuous-time SLAM, see section 3 Since the trajectory of the robot tool is given as transformation matrices with respect to the robot base, an additional transformation from the laser scanner to the tool tip has to be performed.

The application also provides a feature to subsequently distort the path in linear or sinusoidal fashion on different axes, i.e., in six degree of freedom. Those distortions are used to benchmark continuous-time SLAM. This feature is examined in greater detail in section 4

**2.2.4 Time Synchronization** To create an accurate reference 3D point cloud, the two data sets, i.e., laser scanner measurements and tool pose, have to be synchronized precisely. This is achieved by using the NTP protocol, which synchronizes the internal clocks of both machines, so that the ROS nodes are time synchronized. During data acquisition the same time stamps are then used for recording data. So if network delays occur with the packages from the Raspberry Pi to the computer that gathers all data, they are neglectable since the previously recorded ROS time stamps are used. However, further inaccuracies like the RSI timing offset have to be taken into account.

Since the RSI data is processed on the KUKA control unit and multicasted to the network, the tool pose measurements are delayed by a constant offset. To get the accurate timing two algorithms have been developed to detect automatically the beginning and ending of the experiment. Therefore, a short standstill period is needed in the robot arm movement, that can then be detected in the data of the robotic arm joint movements, as well as in the laser recordings. The standstill period of the robot joints is detected by searching for the period where the derivative of the recorded data is zero. For the laser data each profile is compared to the previous one, finding point to point correspondences between the profiles assuming a common frame of reference for both. Then the square-root-distance error between all point pairs is accumulated. After calculating the error for all recorded profiles, the laser data between the two pauses is taken, which is when these accumulated errors sum to zero. Now with both start and stop times for RSI and laser data, the offset can be calculated with significant accuracy. Figure 4 shows a data set with the derivatives of the tool position in all spatial directions and the change of the laser data, using a standstill period of 7 s.

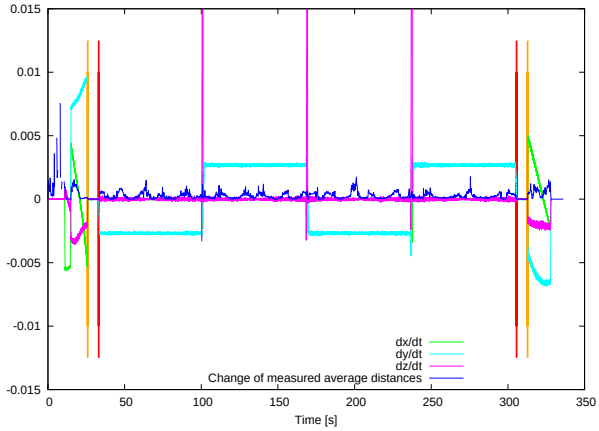


Figure 4. Plot of the inputs used by the synchronization algorithms. The vertical red lines mark the beginning and end of the region of interest.

## CONTINUOUS-TIME SLAM ALGORITHM

### 6D SLAM

To understand the basic idea of the used continuous-time SLAM, we summarize its foundation, 6DSLAM, which was designed for a high-precision registration of terrestrial 3D scans, i.e., globally consistent scan matching (Borrmann et al., 2008). It is available in *3DTK – The 3D Toolkit*. The globally consistent scan matching is an Offline SLAM solution for 3D point clouds, it is a Graph-SLAM algorithm.

6D SLAM works similarly to the well-known iterative closest points (ICP) algorithm, which minimizes the following error function

$$E(\mathbf{R},\mathbf{t}) = \frac{1}{N}\sum_{i=1}^{N} \left\| \mathbf{m}_i - (\mathbf{R}\mathbf{d}_i + \mathbf{t}) \right\|^2 \qquad (1)$$

to iteratively solve for an optimal rotation $\mathbf{T} = (\mathbf{R},\mathbf{t})$, where the tuples $(\mathbf{m}_i, \mathbf{d}_i)$ of corresponding model $\mathbf{M}$ and data points $\mathbf{D}$ are given by minimal distance, i.e., $\mathbf{m}_i$ is the closest point to $\mathbf{d}_i$ within a close limit (Besl and McKay, 1992). Instead of the two-scan-Eq. (1), we look at the $n$-scan case

$$E = \sum_{j\to k}\sum_{i} \left|\mathbf{R}_j\mathbf{m}_i + \mathbf{t}_j - (\mathbf{R}_k\mathbf{d}_i + \mathbf{t}_k)\right|^2, \qquad (2)$$

where $j$ and $k$ refer to scans of the SLAM graph, i.e., to the graph modeling the pose constraints in SLAM or bundle adjustment. If they overlap, i.e., closest points are available, then the point pairs for the link are included in the minimization. We solve for all poses at the same time and iterate like in the original ICP. The derivation of a GraphSLAM method using a Mahalanobis distance that describes the global error of all the poses

$$W = \sum_{j \to k} (\bar{\mathbf{E}}_{j,k} - \mathbf{E}'_{j,k})^T \mathbf{C}_{j,k}^{-1} (\bar{\mathbf{E}}'_{j,k} - \mathbf{E}'_{j,k}) \quad (3)$$
$$= \sum_{j \to k} (\bar{\mathbf{E}}_{j,k} - (\mathbf{X}'_j - \mathbf{X}'_k)) \mathbf{C}_{j,k}^{-1} (\bar{\mathbf{E}}_{j,k} - (\mathbf{X}'_j - \mathbf{X}'_k)).$$

where $\mathbf{E}'_{j,k}$ is the linearized error metric and the Gaussian distribution is $(\bar{\mathbf{E}}_{j,k}, \mathbf{C}_{j,k})$ with computed covariances from scan matching as given in (Borrmann et al., 2008). $\mathbf{X}'_j$ and $\mathbf{X}'_k$ denote the two poses linked in the graph and related by the linear error metric. Minimizing Eq. (2) instead of (3) does not lead to different results (Nüchter et al., 2010). In matrix notation $W$ in Eq. (3) becomes

$$W = (\bar{\mathbf{E}} - \mathbf{H}\mathbf{X})^T \mathbf{C}^{-1} (\bar{\mathbf{E}} - \mathbf{H}\mathbf{X}).$$

Here $\mathbf{H}$ is the signed incidence matrix of the pose graph, $\bar{\mathbf{E}}$ is the concatenated vector consisting of all $\bar{\mathbf{E}}'_{j,k}$ and $\mathbf{C}$ is a block-diagonal matrix comprised of $\mathbf{C}_{j,k}^{-1}$ as submatrices. Minimizing this function yields new optimal pose estimates. Please note, while there are four closed-form solutions for the original ICP Eq. (1), linearization of the rotation in Eq. (2) or (3) seams to be always required.

Globally consistent scan matching is a full SLAM solution for 3D point clouds. It is an offline algorithm and thus optimizes *all* poses in the SLAM pose graph. As a result, processing more scans will increase the algorithm's run-time.

**Continuous-time SLAM**

In previous work, we developed an algorithm that improves the entire trajectory simultaneously. The algorithm is adopted from Elseberg et al. (2013), where it was used in a different mobile mapping context, i.e., on wheeled platforms. Unlike other state of the art algorithms, like (Stoyanov and Lilienthal, 2009) and (Bosse et al., 2012), it is not restricted to purely local improvements. We make no rigidity assumptions, except for the computation of the point correspondences. We require no explicit motion model of a vehicle for instance, thus it works well on backpack systems. The continuous-time SLAM for trajectory optimization works in full 6 DoF. The algorithm requires no high-level feature computation, i.e., we require only the points themselves.

In case of mobile mapping, we do not have separate terrestrial 3D scans. In the current state of the art in the robotics community developed by Bosse et al. (2012) for improving overall map quality of mobile mappers, the time is coarsely discretized. This results in a partition of the trajectory into sub-scans that are treated rigidly. Then rigid registration algorithms like the ICP and other solutions to the SLAM problem are employed. Obviously, trajectory errors within a sub-scan cannot be improved in this fashion. Applying rigid pose estimation to this non-rigid problem directly is also problematic since rigid transformations can only approximate the underlying ground truth. When a finer discretization is used, single 2D scan slices or single points result that do not constrain a 6 DoF pose sufficiently for rigid algorithms.

More mathematical details of the algorithm in the available open-source code and are given in (Elseberg et al., 2013). Essentially, we first split the trajectory into sections, and match these sections using the automatic high-precision registration of terrestrial 3D scans, i.e., globally consistent scan matching that is the 6D SLAM core. Here the graph is estimated using a heuristic that measures the overlap of sections using the number of closest point pairs. After applying globally consistent scan matching on the sections the actual continuous-time or semi-rigid matching as described in (Borrmann et al., 2008; Elseberg et al., 2013) is applied, using the results of the rigid optimization as starting values to compute the numerical minimum of the underlying least square problem. To speed up the calculations, we make use of the sparse Cholesky decomposition by (Davis, 2006).

Given a trajectory estimate, we "unwind" the point cloud into the global coordinate system and use nearest neighbor search to establish correspondences at the level of single scans (those can be single 2D scan profiles). Then, after computing the estimates of pose differences and their respective covariances, we optimize the trajectory. In a predependend step, we consider trajectory elements every $k$ steps and fuse the trajectory elements around these steps $l$ temporarily into a meta-scan.

A key issue in continuous-time SLAM is the search for closest point pairs. We use an octree and a multi-core implementation using OpenMP to solve this task efficiently. A time-threshold for the point pairs is used, i.e., we match only to points if they were recorded at least $t_d$ time steps away. This time corresponds to the number of separate 3D point clouds acquired by the SICK scanner. For most data sets, this was set to 4 sec. ($k = 100$, $l = 200$). In addition, we use a maximal allowed point-to-point-distance which has been set to 25 cm.

Figure 5 shows the trajectory *Circle* that was programmed. It was splitted for GraphSLAM into metascans. Every 100th scan lines a metascan was built with a size of 200 scan lines ($\pm 100$ scan lines). The visualization the multiview orthographic projection and a zoom into the horizontal view. A small shift has been applied to visualize the overlap. Note: only the position of the sensor is plotted, the orientation is not drawn.

After optimization, all scan slices are joined in a single point cloud to enable efficient viewing of the scene. The first frame, i.e., the first 3D scan slice from the SICK scanner defines the arbitrary reference coordinate system of the resulting 3D point cloud.

## BENCHMARKING

The system described in section 2 is used to benchmark the continuous-time SLAM algorithm. A representation of the undistorted laboratory scenery captured with the laser scanner is depicted in figure 6. Numerous data sets have been acquired and then distorted by different methods, processed and corrected. Next, the robotic arm moves the measuring unit through space on three different trajectories. For better readability in further sections each data set is named after the shape of its path. The three trajectories: *Circle*, *Multiline* and *X* are described in the following subsections.

**Robot Path Programming and Data Sets**

Due to the fact that the 3D point clouds are processed with continuous-time SLAM, overlapping profiles of the same area in the point cloud with sufficient quantity of corresponding points are required. To achieve this, and simultaneously obtain a full 3D representation of the environment, the tool tip of the robotic arm

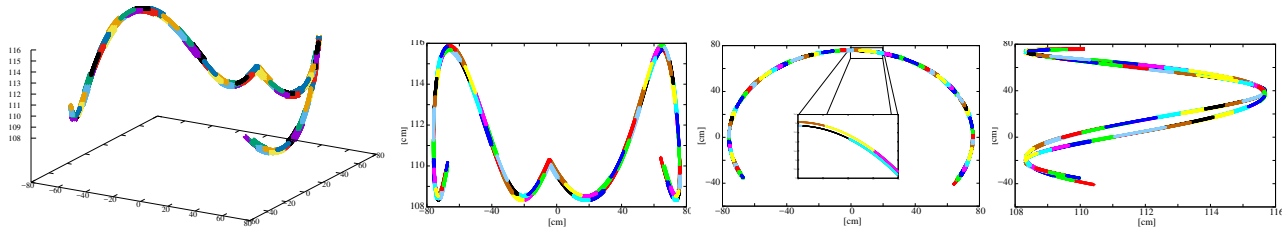

Figure 5. Trajectory describing a circular arc in the horizontal plane. Cf. Figure 7. From left to right: 3D view and detailed view. Vertical, horizontal, and vertical orthographic projection.

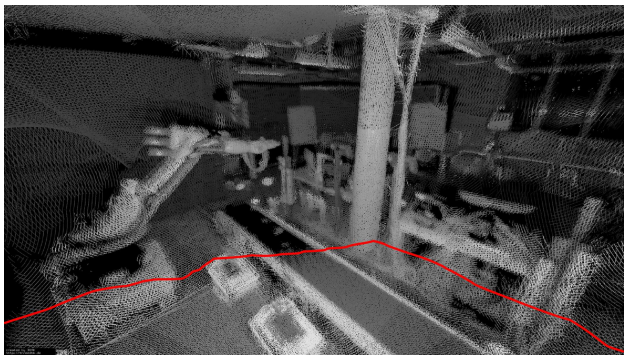

Figure 6. 3D point cloud of the laboratory. In the foreground, the trajectory is shown in red.

is rotated along its roll axis multiple times back and forth. For that motion the spherical wrist of the robot is used, while the remaining axes are deployed to move the robotic arm on its path. A uniform rotary motion in just one direction of more than 480 ° is not feasible because of joint angle restrictions and winding cables of the gripper. In general, the boundaries of the chosen path are limited by the physical reachable positions and the configuration singularities of the robotic arm.

For data set *Circle* the measuring unit is moved on a circular path around the base of the robotic arm with a constant height, cf. figure 5. During that motion the spherical wrist is used to make four 480 ° rotations, where the direction of rotation is inverted each time.

Data set *Multiline* is acquired by moving the measuring unit on a straight - horizontal line at constant speed while simultaneously rotating the scanner in one direction for 450 °. Due to restrictions in the arm configuration, the maximum of 480 ° cannot be achieved by this trajectory. This motion is repeated four times along the same axis while the measuring unit is raised by 20 cm at the end of each horizontal segment. Also, the direction of rotation and translation is inverted after each segment.

The processed point cloud of the last data set resembles an *X* shape that is applied here for naming conventions instead of the shape of the path. The trajectory used for data set *X* also moves the measuring unit on a straight-horizontal line at a constant speed. However, the measuring unit is not rotated during this motion but solely tilted with a fixed angle of 45 °. This motion is repeated two times back and forth along the same axis without any change in height. Before the start of the next segment, the measuring unit is rotated by 90 ° to create a slight overlap in the recorded profiles. The purpose of this trajectory is to show the effects of the correction algorithm when little or no overlap in the recorded data is present.

**Path Distortions and Results**

To verify the performance of the continuous-time SLAM algorithm, the recorded path is distorted by adding different offsets to the tool pose to simulate noise. Afterwards the resulting point cloud is corrected using the algorithm and the results are compared to the original data. Two types of distortions are applied: a constant offset that simulates an incremental error over time, and a sinusoidal offset that is added to the path to one axis with a constant wave number and amplitude. The parameters for the applied distortions as well as for the continuous-time SLAM algorithm are specified in Table 1.

To verify the performance of continuous-time SLAM, CloudCompare (Girardeau-Montaut, 2017) was used to analyze changes in the resulting point clouds. The originally recorded data set and the corrected point cloud are illustrated as heat maps and colored with reflectance values of the laser scanner. The originally recorded data set is improved by using continuous-time SLAM. This is done to remove sparse artifacts and misalignments due to calibration errors, in order to enhance the consistency of the point cloud. Further evaluations are based on this reference point cloud. A comparison is established by superimposing the reference and a corrected point cloud, visualizing the absolute error in a separate heat map.

Also, the corrected path is evaluated under two aspects: First by comparing the absolute point-to-point distance, and second by the change in orientation of the measuring unit in contrast to the original path.

**4.2.1 Data Set: *Circle*** Figure 7 shows on the left side the originally acquired and processed point cloud. The middle image shows the resulting point cloud after processing the recorded data with continuous-time SLAM. This point cloud is used as a reference to calculate the error after correcting a distorted version of the original data. Figure 8 shows the results of the applied linear distortion. The maximum error compared to the reference after correcting the distortion is 0.02 m as shown in the top right picture. The maximum point to point error of the trajectory is reduced from 1 m distortion to 0.14 m, the orientation is changed by a maximum of 4.5 °. Figure 9 shows the results of the applied sinusoid distortion. After 200 iterations, the errors are clearly reduced but not fully corrected. The maximum point to point error is 0.25 m but also note that the error distribution shown at the right of the error color bar indicates that the majority of points have a point to point error that rarely exceeds 0.03 m. The highly disturbed trajectory has a maximum distance of 0.25 m while the orientation is changed by a maximum of 7 °.

**4.2.2 Data Set: *Multiline*** Figure 10 shows on the left side the original data recording from the measuring unit. The middle image shows the corrected point cloud, to eliminate any existing misalignments. For this data set, this is the point cloud that is used as comparison reference for the corrected point clouds after

| Dataset | Profiles | Distortion | Amplitude [m] | Wavenumber | Metascan | Overlap | Iterations Iterations | Duration [min] |
|---|---|---|---|---|---|---|---|---|
| Circle | 15,194 | None | — | — | 100 | 200 | 200 | 2071 |
| | | Linear | 1 | — | 100 | 200 | 200 | 2125 |
| | | Sinusoid | 0.2 | 12.5 | 100 | 200 | 200 | 2125 |
| Multiline | 12,321 | None | — | — | 400 | 400 | 200 | 388 |
| | | Linear | 1 | — | 400 | 400 | 200 | 379 |
| | | Sinusoid 1 | 0.2 | 20 | 400 | 400 | 200 | 440 |
| | | Sinusoid 2 | 0.2 | 40 | 400 | 400 | 200 | 437 |
| X | 8400 | Linear | 0.5 | — | 4200 | 4200 | 50 | 1 |
| | | Linear | 0.5 | — | 250 | 500 | 100 | 24 |

Table 1. Parameters for applied distortions and continuous-time SLAM.

being distorted. Figure 11 presents the results after applying a linear distortion. The maximum error after correcting the point cloud compared to the reference is 0.04 m as shown in the top right picture. The maximum point to point error of the trajectory is reduced from 1 m distortion to 0.045 m, the orientation is changed by a maximum of 0.45 °. Figure 12 shows the result of applying a sinusoid distortion with an amplitude of 0.2 m and a wavenumber of 20. The corrected data, shown in the middle, is then compared to the reference frame to calculate the point to point error, shown on the right. The maximum point to point error is 0.3 m for this case, but also note the error distribution, which shows that the majority of points rarely have point to point errors greater than 0.03 m. The disturbed trajectory has a maximum distance of 0.45 m while the orientation changed by a maximum of 3 °. Finally Figure 13 shows the output of applying another sinusoid distortion but with a wavenumber of 40. The middle image shows the corrected data, and this is then compared to the reference frame and the resulting point to point error is shown in the right image. The resulting maximum error is 0.135 m, and analysing the error distribution, most points do not have point to point error greater than 0.04 m. The disturbed trajectory has a maximum distance of 0.2 m while the orientation changed by a maximum of 2 °.

**4.2.3 Data Set: *X*** Figure 14 shows on the left side the original point cloud from the laser scanner. Due to the fact that the scanner does not rotate, overlap is only achieved by the direction change of the scanner. As a consequence, consecutive profiles have zero overlap between them, in contrast to the previous trajectories. The middle image shows the approach to correct a linear distortion with carefully chosen parameters, but almost no improvement is visible. The right image shows the same approach with different parameters, but due to no overlap between consecutive metascans, the point cloud breaks.

## CONCLUSIONS AND FUTURE WORK

This paper show a novel way to benchmark SLAM algorithms based on a ground truth acquired from the motion of a robotic arm. It can be used as a tool to show the influence of arbitrary forms of noise on the ground truth trajectory and the capabilities of the applied SLAM algorithm. The presented system enables us, to systematically evaluate all six degree of freedom of a mobile mapping system, which was not possible in the past, with vehicle-based mobile mapping systems. The setup enables us to compare 3D point clouds, but also sensor positions and orientations. The evaluation shows, that careful selection of parameters is needed to enable the cenvergance to the global minimum.

Needles to say, a lot of work remains to be done. First of all, as calibration is crucial for SLAM, the accuracy of the benchmarking facility can be further improved. For instance, by determining a more precise coordinate transformation between the coordinate systems of the laser scanner reference frame and the tool reference frame of the robotic arm.

Furthermore, since even small timing errors induce inaccuracies in the resulting point cloud, time synchronization between the laser scanner frames and the robot pose remains an essential aspect. Direct access to the control unit of the robotic arm minimizes this delay. Additionally, a system that triggers the start and stop times of the benchmarking experiment is useful.

Moreover, attaching the measuring unit directly to the robotic arm, thus omitting the gripper, would allow for more rotatability and flexibility and reduce the number of erroneous influences.

## References

Anderson, S., MacTavish, K. and Barfoot, T. D., 2015. Relative continuous-time slam. *International Journal of Robotics Research (IJRR)* 34(12), pp. 1453–1479.

Andreas Nüchter et al., 2017. 3DTK – The 3D Toolkit. `http://slam6d.sourceforge.net/`.

Barfoot, T. D., Tong, C. H. and Särkka, S., 2014. Batch continuous-time trajectory estimation as exactly sparse gaussian process regression. In: *Proceedings of Robotics: Science and Systems (RSS '14)*, Berkeley, CA, USA.

Besl, P. and McKay, N., 1992. A Method for Registration of 3–D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 14(2), pp. 239–256.

Borrmann, D., Elseberg, J., Lingemann, K., Nüchter, A. and Hertzberg, J., 2008. Globally consistent 3d mapping with scan matching. *Journal Robotics and Autonomous Systems (JRAS)* 56(2), pp. 130–142.

Bosse, M., Zlot, R. and Flick, P., 2012. Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping. *IEEE Transactions on Robotics (TRO)* 28(5), pp. 1104–1119.

Davis, T. A., 2006. Direct Methods for Sparse Linear Systems. SIAM.

Elseberg, J., Borrmann, D. and Nüchter, A., 2013. Algorithmic solutions for computing accurate maximum likelihood 3D point clouds from mobile laser scanning platforms. *Remote Sensing* 5(11), pp. 5871–5906.

Girardeau-Montaut, D., 2017. Cloudcompare. `http://www.cloudcompare.org/`.

Hess, W., Kohler, D., Rapp, H. and Andor, D., 2016. Real-time loop closure in 2d lidar slam. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1271–1278.

Houshiar, H., Elseberg, J., Borrmann, D. and Nüchter, A., 2015. A Study of Projections for Key Point Based Registration of Panoramic Terrestrial 3D Laser Scans. *Journal of Geo-spatial Information Science* 18(1), pp. 11–31.

Kaul, L., Zlot, R. and Bosse, M., 2016. Continuous-time three-dimensional mapping for micro aerial vehicles with a passively actuated rotating laser scanner. *Journal of Field Robotics* 33(1), pp. 103–132.

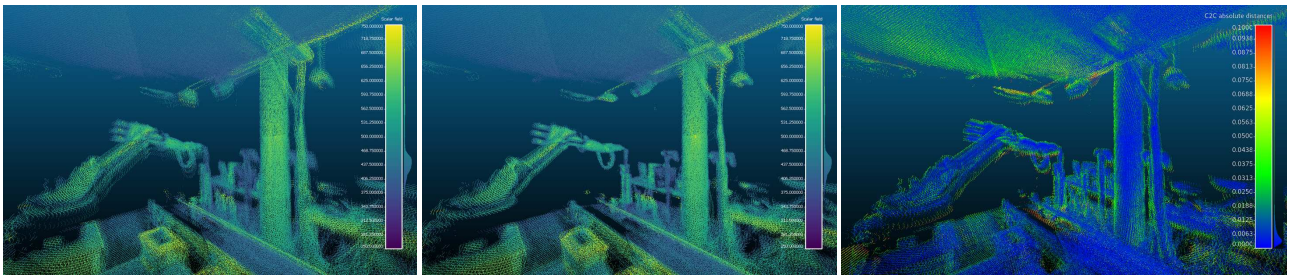

Figure 7. 3D point cloud of the laboratory. Left: Recorded data. Middle: Corrected point cloud used as reference. Right: Error between recorded and corrected point clouds.

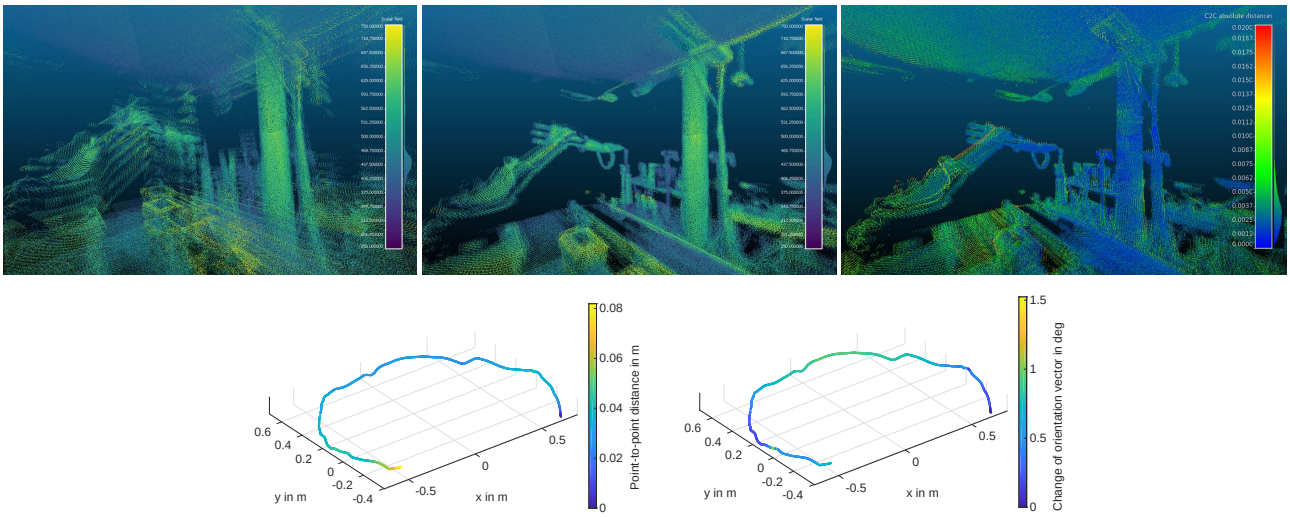


Figure 8. Linear Distortion: Top Left: Input point cloud. Top Middle: Corrected point cloud. Top Right: Error between corrected and reference point cloud. Bottom Left: Point to point distance between the corrected and original trajectory. Bottom Right: Change of orientation vector between the corrected and original trajectory.

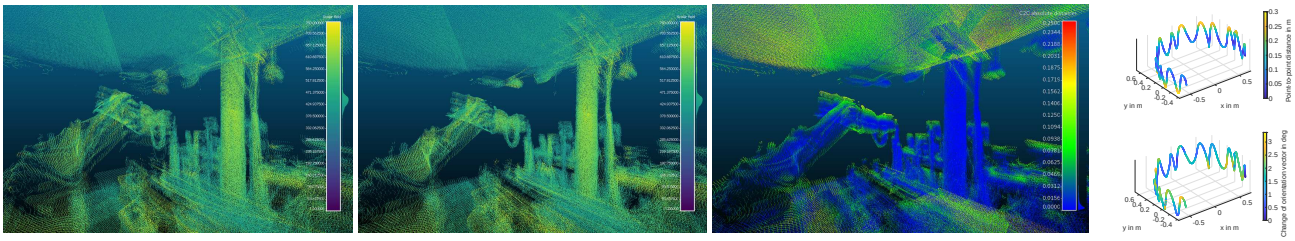

Figure 9. Sinusoid Distortion: Left: Input Point Cloud. Middle: Corrected point cloud. Right: Error between corrected and reference point cloud. Plots: Above: Point to point distance between the corrected and original trajectory. Below: Change of orientation vector between the corrected and original trajectory.

Lehtola, V. V., Virtanen, J.-P., Vaaja, M. T., Hyyppä, H. and Nüchter, A., 2016. Localization of a mobile laser scanner via dimensional reduction. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)* 121, pp. 48–59.

Moosmann, F. and Stiller, C., 2011. Velodyne SLAM. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV '11)*, Baden-Baden, Germany, pp. 393–398.

Nüchter, A., Borrmann, D., Koch, P., Kühn, M. and May, S., 2015. A man-portable, imu-free mobile mapping system. In: *Proceedings of the ISPRS Geospatial Week 2015, Laserscanning 2015*, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., II-3/W5, La Grande Motte, France, pp. 17–23.

Nüchter, A., Elseberg, J., Schneider, P. and Paulus, D., 2010. Study of Parameterizations for the Rigid Body Transformations of The Scan Registration Problem. *Journal Computer Vision and Image Understanding (CVIU)* 114(8), pp. 963–980.

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A. Y., 2009. Ros: an open-source robot operating system. In: *ICRA Workshop on Open Source Software*.

Schwertfeger, S. and Birk, A., 2013. Evaluation of Map Quality by Matching and Scoring High-Level, Topological Map Structures. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '13)*.

Schwertfeger, S., Jacoff, A. S., Scrapper, C., Pellenz, J. and Kleiner, A., 2011. Evaluation of Maps using Fixed Shapes: The Fiducial Map Metric. In: *Proceedings of the 2010 Performance Metrics for Intelligent Systems (PerMIS '10) Workshop*, NIST, Baltimore, MD, USA.

Stoyanov, T. and Lilienthal, A. J., 2009. Maximum Likelihood Point Cloud Acquisition from a Mobile Platform. In: *Proceedings of the IEEE International Conference on Advanced Robotics (ICAR '09)*, pp. 1–6.

Wulf, O., Nüchter, A., Hertzberg, J. and Wagner, B., 2008. Benchmarking Urban 6D SLAM. *Journal of Field Robotics (JFR)* 25(3), pp. 148–163.

Zhang, J. and Singh, S., 2014. LOAM: Lidar Odometry and Mapping in Real-time. In: *Proceedings of Robotics Science and Systems (RSS '14)*, Berkeley, CA, USA.

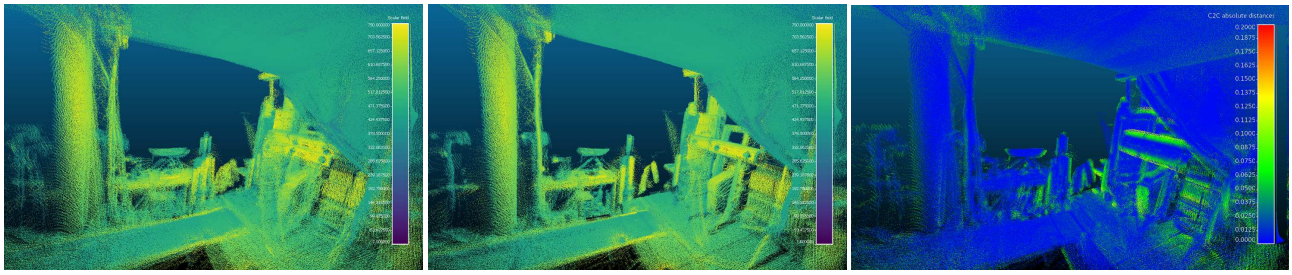

Figure 10. 3D point cloud of the laboratory. Left: Recorded data. Middle: Corrected point cloud used as reference. Right: Error between recorded and corrected point clouds.

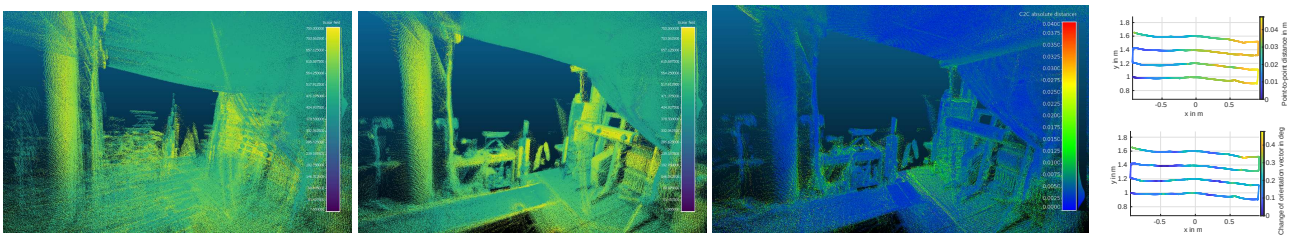

Figure 11. Linear Distortion: Left: Input point cloud. Middle: Corrected point cloud. Right: Error between corrected and reference point cloud. Plots: Above: Point to point distance between the corrected and original trajectory. Below: Change of orientation vector between the corrected and original trajectory.

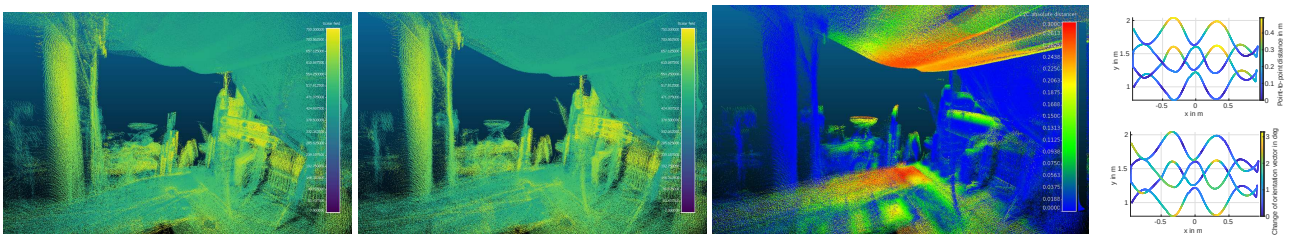

Figure 12. Sinusoid Distortion: Left: Input Point Cloud. Middle: Corrected point cloud. Right: Error between corrected and reference point cloud. Plots: Above: Point to point distance between the corrected and original trajectory. Below: Change of orientation vector between the corrected and original trajectory.

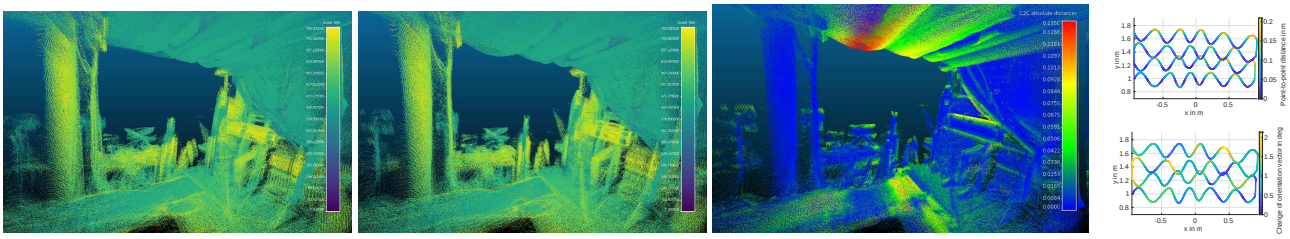

Figure 13. Sinusoid Distortion: Left: Input Point Cloud. Middle: Corrected point cloud. Right: Error between corrected and reference point cloud. Plots: Above: Point to point distance between the corrected and original trajectory. Below: Change of orientation vector between the corrected and original trajectory.

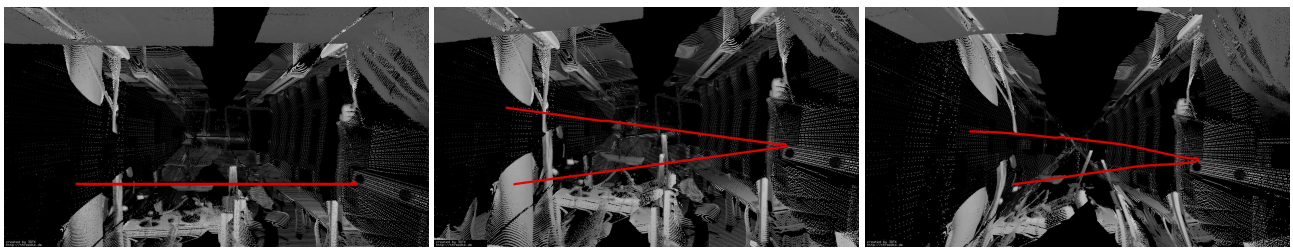

Figure 14. Linear trajectory without overlap between consecutive scans. Left: Original data as recorded. Middle: Corrected point cloud with carefully selected parameters after applying a linear distortion. Right: Corrected point cloud with standard parameters.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
doi:10.5194/isprs-annals-IV-2-W4-91-2017 | © Authors 2017. CC BY 4.0 License.